Problem Set 9

This problem set is due **at 11:59pm** on **Thursday, April 30, 2015.**

Each submitted solution should start with your name, the course number, the problem number, your recitation section, the date, and the names of any students with whom you collaborated.

Exercise 9-1. Read CLRS, Chapter 35.

Exercise 9-2. Exercise 35.2-3.

Exercise 9-3. Exercise 35.4-2.

Exercise 9-4. Read Prof. Demaine's notes on fixed-parameter algorithms.

Problem 9-1. Knapsack [25 points]

In the Knapsack problem, we are given a set $A = \{a_1, \ldots, a_n\}$ of items, where each a_i has a specified positive integer *size* s_i and a specified positive integer *value* v_i. We are also given a positive integer *knapsack capacity* B. Assume that $s_i \leq B$ for every i. The problem is to find a subset of A whose total size is at most B and for which the total value is maximized.

In this problem, we will consider approximation algorithms to solve the Knapsack problem.

Notation: For any subset S of A, we write s_S for the total of all the sizes in S and v_S for the total of all the values in S. Let Opt denote an optimal solution to the problem.

(a) [5 points] Consider the following greedy algorithm Alg_1 to solve the Knapsack problem:
Order all the items a_i in non-increasing order of their *density*, which is the ratio of value to size, $\frac{v_i}{s_i}$. Make a single pass through the list, from highest to lowest density. For each item encountered, if it still fits, include it, otherwise exclude it.

Prove that algorithm Alg_1 does not guarantee any constant approximation ratio. That is, for any positive integer k, there is an input to the algorithm for which the total value of the set of items returned by the algorithm is at most $\frac{v_{Opt}}{k}$.

(b) [7 points] Consider the following algorithm Alg_2.
If the total size of all the items is $\leq B$, then include all the items. If not, then order all the items in non-increasing order of their densities. Without loss of generality, assume that this ordering is the same as the ordering of the item indices. Find the smallest index i in the ordered list such that the total size of the first i items exceeds B (i.e.,

$\sum_{j=1}^{i} s_j > B$, but $\sum_{j=1}^{i-1} s_j \leq B$). If $v_i > \sum_{j=1}^{i-1} v_j$, then return $\{a_i\}$. Otherwise, return $\{a_1, \ldots, a_{i-1}\}$.

Prove that Alg_2 always yields a 2-approximation to the optimal solution.

(c) [5 points] Let $A = \{a_1, \ldots, a_n\}$ be the input ordered arbitrarily, and V be the largest value for any item; then nV is an upper bound on the total value that could be achieved by any solution. For every $i \in \{1, \ldots, n\}$ and $v \in \{1, \ldots, nV\}$, define $S_{i,v}$ to be the smallest total size of a subset of $\{a_1, \ldots, a_i\}$ whose total value is exactly v; $S_{i,v} = \infty$ if no such subset exists.

Give a dynamic programming algorithm Alg_3 to solve Knapsack exactly. Specifically, give a recurrence to compute all values of $S_{i,v}$, and explain how to use this to solve the Knapsack problem. Analyze its time complexity.

(d) [8 points] Finally, we develop Alg_4, which is a *Fully Polynomial Time Approximation Scheme (FPTAS)* for Knapsack. The idea is to use an exact dynamic programming algorithm like the one in Part (c), but instead of using the given (possibly large) values for the items, we use versions of the given values that are suitably scaled and rounded down. As in Part (c), order $A = \{a_1, \ldots, a_n\}$ arbitrarily, and let V be the largest value for any item.

For any $\varepsilon, 0 < \varepsilon < 1$, Alg_4 behaves as follows:
For each item a_i with value v_i, define a scaled value $v_i' = \lfloor (\frac{v_i}{V})(\frac{n}{\varepsilon}) \rfloor$.
Using these scaled values (and the given sizes), run Alg_3 and output the set C of items that it returns.

Prove that Alg_4 is a FPTAS for Knapsack.

Problem 9-2. Fixed-Parameter Algorithms [25 points]

We consider the *Tournament Edge Reversal* problem. Define a ***tournament*** to be a directed graph $T = (V, E)$ such that, for every pair of vertices $u, v \in V$, exactly one of (u, v) and (v, u) is in E. Furthermore, define a ***cycle cover*** to be a set $A \subset E$ of directed edges of a tournament T such that, every directed cycle of T contains at least one edge from A.

(a) [5 points] Let a minimal cycle cover of T be a cycle cover with the least number of edges. Prove that reversing all the edges of a minimal cycle cover A turns T into an acyclic tournament. (*Hint:* Any edge $e \in A$ must be the *only* edge in A on some directed cycle of T.)

In the Tournament Edge Reversal problem, we are given a tournament T and a positive integer k, and the objective is to decide whether T has a cycle cover of size at most k.

(b) [15 points] Show that this problem has a kernel with at most $k^2 + 2k$ vertices. (*Hint:* Define a ***triangle*** to be a directed cycle of length 3. Consider the number of times that a node or an edge appears in different triangles.)

(c) [5 points] Obtain a FPT algorithm for the Tournament Edge Reversal problem.

6.046J / 18.410J Design and Analysis of Algorithms
Spring 2015